Data Mining Exam

April 23, 2021

Kasper Rosenkrands

Aalborg University
Denmark

AALBORG UNIVERSITY
DENMARK

Agenda

Clustering
Introduction

Clustering is a way to categorize data to impose structure.

A use case is recommender systems (Amazon, Spotify, Netflix), where a user is recommended items that bought/listened to/watched by other users with similar interests.

Clustering

K-Means Optimization Problem

Given $D = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^p$, $K \in \mathbb{N}$ and let $C_1, \ldots, C_K$ denote different groups of the x_i's.

The K-Means algorithm tries to solve

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}, \tag{1}$$

where $W(C_k)$ denotes the **within cluster variation**, in other words the dissimilarity of the group.

The most common dissimilarity measure is the is the squared Euclidean distance

$$W(C_k) := \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{i,j} - x_{i',j})^2. \tag{2}$$

Clustering
K-Means Optimization Problem

If we by $\bar{x}_{k,j} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{i,j}$ denote the mean value of the j'th dimension in cluster k, it can be shown that

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{i,j} - x_{i',j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - \bar{x}_{k,j})^2. \tag{3}$$

If we further note that $\bar{x}_{k,j} = \min_{\mu_k} \left\{ \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - \mu_k)^2 \right\}$ this implies that the optimization problem in (1) can be rewritten as

$$\min_{C_1,\dots,C_K,\mu_1,\dots,\mu_K} \left\{ \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - \mu_k)^2 \right\}. \tag{4}$$

Clustering
K-Means Algorithm

The K-Means algorithm is now able to exploit the new formulation of the optimization problem and iteratively solve for $\{C_1, \ldots, C_k\}$ and $\{\mu_1, \ldots, \mu_k\}$.

This makes K-Means a greedy algorithm because, in each iteration it chooses optimal values for $\{C_1, \ldots, C_k\}$ and $\{\mu_1, \ldots, \mu_k\}$.

Convergence of the algorithm is therefore ensured, however we cannot guarantee it will find the global optimum.

Clustering

K-Means Algorithm

Algorithm 1: K-Means

```
Assign each obsevation to a cluster randomly
foreach Cluster do
    Compute the centroid
foreach Observation do
    Compute distance to all centroids
    Assign to the closest
while Centroids have not changed since last iteration do
    foreach Observation do
        Compute distance to all centroids
        Assign to the closest
    foreach Cluster do
        Compute the centroid
return Clusters
```

Clustering

An example of the K-Means algorithm

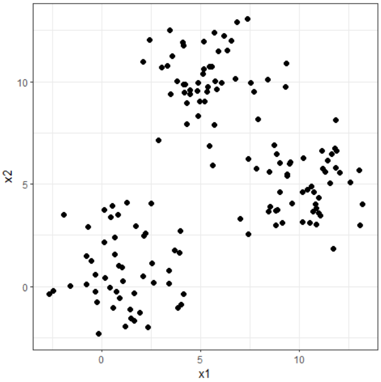

Figure: Iteration 01

Clustering

An example of the K-Means algorithm

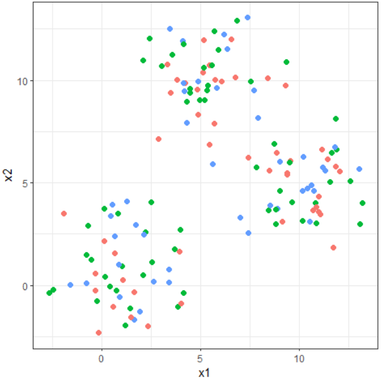

Figure: Iteration 02

Clustering

An example of the K-Means algorithm

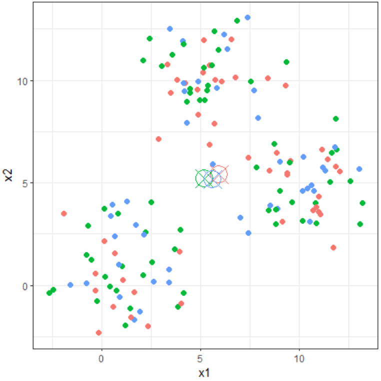

Figure: Iteration 03

Clustering
An example of the K-Means algorithm

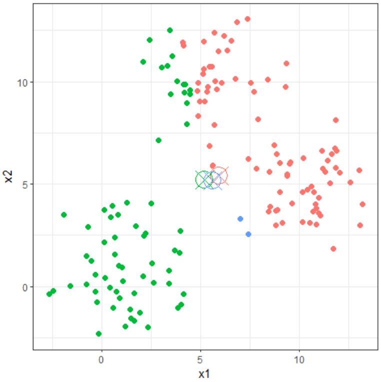

Figure: Iteration 04

Clustering

An example of the K-Means algorithm

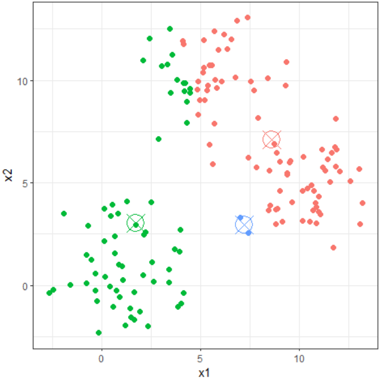

Figure: Iteration 05

Clustering

An example of the K-Means algorithm

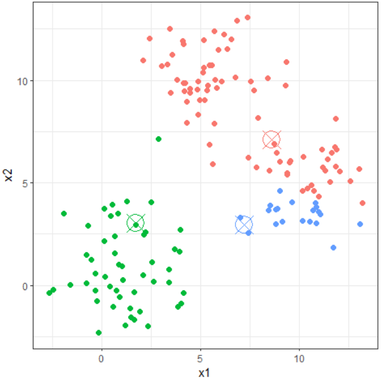

Figure: Iteration 06

Clustering

An example of the K-Means algorithm

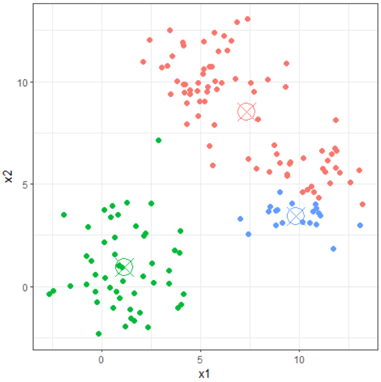

Figure: Iteration 07

Clustering

An example of the K-Means algorithm

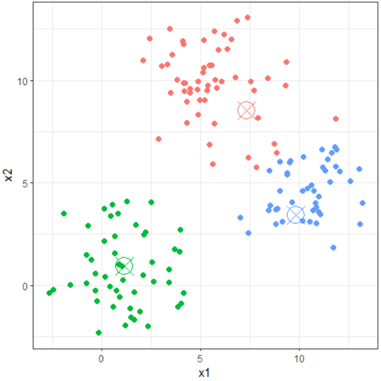

Figure: Iteration 08

Clustering

An example of the K-Means algorithm

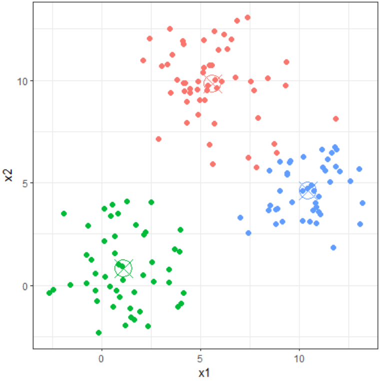

Figure: Iteration 09

Clustering

An example of the K-Means algorithm

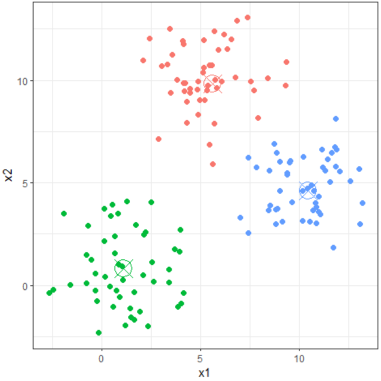

Figure: Iteration 10

Clustering

An example of the K-Means algorithm

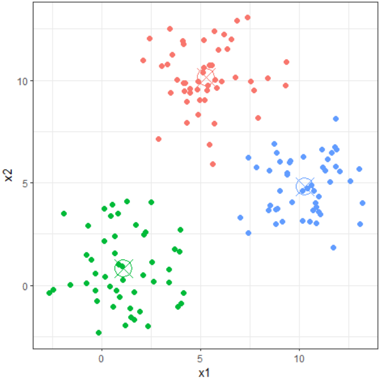

Figure: Iteration 11

Clustering

An example of the K-Means algorithm

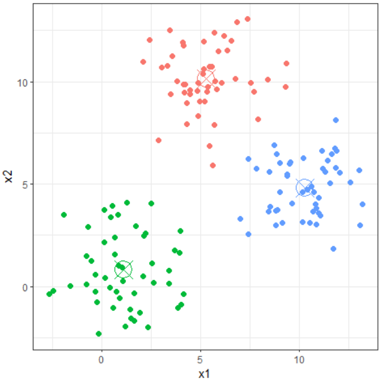

Figure: Iteration 12

Clustering

An example of the K-Means algorithm

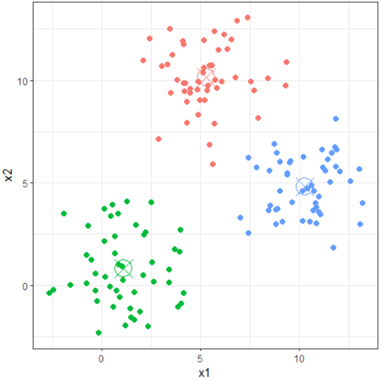

Figure: Iteration 13

Clustering
Hierarchical Clustering

Shrinkage
Lasso

Shrinkage
Ridge Regression

Shrinkage

Elastic Net

Classification

Linear Discriminant Analysis (LDA)

Classification

Quadratic Discriminant Analysis (QDA)

Classification
Naive Bayes

Trees

Classification and Regression Trees (CART)

Trees
Bagging

Trees
Random Forest

Trees
Boosting

Support Vector Machines

Neural Networks
Perceptron

Backpropagation: For a given loss function L we look for $\frac{\partial L}{\partial w_i}$. We start with initial values for the weights, which we shall denote w_{old}. Then we update the weights by $w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$. One iteration is called and **epoch**.